UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/699,486 | 10/31/2003 | Eric Anderson | 200207252-1 | 3149 |

22879 7590 06/13/2008

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

| EXAMINER | |
|---|---|
| RADTKE, MARK A | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2165 | |

| NOTIFICATION DATE | DELIVERY MODE |
|---|---|
| 06/13/2008 | ELECTRONIC |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

JERRY.SHORMA@HP.COM
mkraft@hp.com
ipa.mail@hp.com

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

**BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

Application Number: 10/699,486
Filing Date: October 31, 2003
Appellant(s): ANDERSON, ERIC

Philip S. Lyren
For Appellant

**EXAMINER'S ANSWER**

This is in response to the appeal brief filed 31 March 2008 appealing from the Office action mailed 30 October 2007.

**(1) Real Party in Interest**

A statement identifying by name the real party in interest is contained in the brief.

**(2) Related Appeals and Interferences**

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

**(3) Status of Claims**

The statement of the status of claims contained in the brief is correct.

**(4) Status of Amendments After Final**

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

**(5) Summary of Claimed Subject Matter**

The summary of claimed subject matter contained in the brief is correct.

**(6) Grounds of Rejection to be Reviewed on Appeal**

The appellant's statement of the grounds of rejection to be reviewed on appeal is substantially correct. The changes are as follows: there is no claim 24. Claims 1-2, 4-15

and 17-24 are rejected under Verma in view of Berliner, and further in view of Deshayes. Claim 16 is rejected under Verma, as modified, and further in view of Kung.

**(7) Claims Appendix**

The copy of the appealed claims contained in the Appendix to the brief is correct.

**(8) Evidence Relied Upon**

| | | |
|---|---|---|
| 6,856,993 | VERMA | 2-2005 |
| 6,047,294 | DESHAYES | 4-2000 |

B. Berliner "CVS II: Parallelizing Software Development" Proceedings of the USENIX Winter 1990 Technical Conference, 1990.

Kung, H.T. "On optimistic methods for concurrency control" ACM Transactions on Database Systems (TODS), vol2, no. 2 (June 1981), pp. 213-226

**(9) Grounds of Rejection**

The following ground(s) of rejection are applicable to the appealed claims:

***Claim Rejections - 35 USC § 103***

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the

invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 1-2, 4-15, and 17-24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Verma et al. (U.S. Pat. No. 6,856,993), in view of Berliner ("CVS II: Parallelizing Software Development" by B. Berliner, Proceedings of the USENIX Winter 1990 Technical Conference, available online at http://citeseer.ist.psu.edu/berliner90cvs.html) and further in view of Deshayes (U.S. Pat. No. 6,047,294).

As to claim 1, Verma et al. teaches a method of creating a filesystem with transaction based functionality (see Abstract), comprising:

receiving an indicator to initiate a transaction for files stored in one or more portions of the filesystem (see column 10, lines 8-10, "mark the thread/process as transacted" and column 10, lines 20-24, "copyFile");

processing the text-based commands written to the control text file (see column 2, lines 57-59 and column 3, lines 3-6); and

operating on one or more portions of the pseudo-filesystem within a transaction according to the text-based commands (see column 3, lines 3-6).

Verma et al. does not explicitly teach creating a control text file that provides a textual filesystem interface and receives text-based commands to operate on the pseudo-filesystem.

However, Berliner teaches creating a control text file that provides a textual filesystem interface and receives text-based commands to operate on the pseudo-filesystem (See section 2.2, "Tracking Third-Party Source Distributions", "**checkin** program". Berliner anticipates the use of scripts, which are equivalent to "a control text file". Furthermore, the use of scripts to automate certain tasks is extremely well-known in the art of Unix systems programming. See, for example, "Running Arbitrary Scripts Under CVS" by J. Vesperman. Furthermore, to an application, the Unix command line ("stdin") is indistinguishable from a text file).

Therefore, it would have been obvious to one having ordinary skill in the relevant art at the time the invention was made to have modified Verma et al. by the teaching of Berliner because "other operating systems and/or file systems may implement and benefit from the present invention" (see Verma et al., column 6, lines 17-19).

Verma et al., as modified, still does not teach duplicating the filesystem within a pseudo-filesystem.

However, Deshayes teaches duplicating the filesystem within a pseudo-filesystem (see Abstract, "a virtual disk partition, may be backed up at a physical level from a primary storage device").

Therefore, it would have been obvious to one of ordinary skill in the relevant art at the time the invention was made to have further modified Verma et al., as modified, by the teaching of Deshayes because all the claimed elements were known in the prior art and one skilled in the art could have combined by known methods with no change in their respective functions, and the combination would have yielded predictable results to

one of ordinary skill in the art at the time the invention was made. Furthermore, it would provide the benefit of solving the problem of having "to change the operating system, or the application programs, every time a change is made to physical storage" (see Deshayes, column 1, lines 30-32).

As to claim 2, Verma et al., as modified, teaches wherein the duplicating is performed lazily (see column 2, lines 59-65 and column 23, "Deferred Redo Alternative") in order to reduce processing impact on the filesystem (This portion of the claim is considered "intended use" and will not be given patentable weight. The effect of "reducing processing impact" is merely a benefit of using the invention and does not functionally relate to the claimed invention).

As to claim 4, Verma et al., as modified, teaches further comprising:

completing the transaction upon receipt of a text-based command associated with terminating the transaction (see column 8, lines 26-28).

As to claim 5, Verma et al., as modified, teaches wherein the text-based commands include functional equivalent commands associated with terminating the transaction (see column 7, lines 23-26, "aborted") and selected from a set of commands for performing one of the following functions: delete directory (see column 17, lines 3-7), delete filesystem (see column 17, lines 3-7, "recursive delete"), and abort (see column 7, lines 23-26).

As to claim 6, Verma et al., as modified, teaches further comprising:

updating the filesystem with updates performed on the pseudo-filesystem when the transaction has completed (see column 8, lines 26-28).

As to claim 7, Verma et al., as modified, teaches wherein the updates are performed upon receipt of an indication to commit the transaction (see column 8, lines 26-28).

As to claim 8, Verma et al., as modified, teaches further comprising:

creating a status text file that provides text-based status results from operations performed on the pseudo-filesystem (see column 2, lines 57-59, "actual data write details of the transaction").

As to claim 9, Verma et al., as modified, teaches wherein the indicator to initiate the transaction results from the creation of a directory within the pseudo-filesystem (see column 27, lines 64-67).

As to claim 10, Verma et al., as modified, teaches wherein the transaction ensures atomic updates to the filesystem in accordance with modifications made to the pseudo-filesystem and related files during the transaction (see column 6, lines 24-26).

As to claims 11 and 18, Verma et al., as modified, teaches wherein a user assists in reconciliation of conflicts between updates in the pseudo-filesystems (See column 29, lines 37-45. Depending on when the non-transacted user releases the resource, a file handle in conflict will not be deleted, thus resolving a resource conflict).

As to claim 12, Verma et al. teaches a method of interfacing with a filesystem (see Abstract) comprising:

For the remaining steps of this claim appellant(s) is/are directed to the remarks and discussions made in claims 1 and 8 above.

As to claim 13, Verma et al., as modified, teaches creating an entire copy of the filesystem (See Examiner's comments regarding claim 1. See Deshayes, Abstract, "a virtual disk partition, may be backed up at a physical level from a primary storage device");

mounting the entire copy of the filesystem under the pseudo-filesystem (see Deshayes, column 12, lines 54-58, "mounting or importing virtual volumes").

As to claim 14, Verma et al., as modified, teaches creating a textual interface;

receiving the text-based command from a user into the textual interface (see column 10, lines 23-24, "command line batch scripts").

As to claim 15, Verma et al., as modified, teaches wherein receiving a text-based command includes functional equivalent commands selected from a set including: change root directory (The "mount" command is all well-known command in NTFS. Mount points can be partitions or folders within an existing partition. See http://support.microsoft.com/?kbid=205524), select concurrency control type (See column 6, lines 56-59. Any kind of concurrency control system can be used via interfaces), select isolation level (See column 6, lines 48-51. Processes, file handles or files must be selected before they are treated as transactional operations. Disabling or enabling transactions is a selection of isolation level.), commit transaction (see column 8, lines 26-28), and abort transaction (see column 7, lines 23-26).

As to claim 17, Verma et al., as modified, teaches wherein determining the one or more data dependencies includes using lock-based concurrency control (LBCC) to control pending read and write operations to the pseudo-filesystem, the filesystem and one or more corresponding files associated with the pseudo-filesystem and filesystem respectively (see column 11, line 49 – column 12, line 18).

As to claim 19, Verma et al. teaches a computer program product for creating a filesystem with transaction based functionality, tangibly stored on a computer-readable medium, comprising instructions operable to cause a programmable processor (see Abstract) to:

For the remaining steps of this claim appellant(s) is/are directed to the remarks and discussions made in claim 1 above.

As to claim 20, Verma et al. teaches a computer program product for interfacing with a filesystem, tangibly stored on a computer-readable medium, comprising instructions operable to cause a programmable processor (see Abstract) to:

For the remaining steps of this claim appellant(s) is/are directed to the remarks and discussions made in claim 1 above.

As to claim 21, Verma et al. teaches an apparatus that creates a filesystem with transaction based functionality (see Abstract) comprising:

a processor (see figure 1, element 21);

a memory (see figure 1, element 25) having instructions capable of being executed on the processor...

For the remaining steps of this claim appellant(s) is/are directed to the remarks and discussions made in claim 1 above.

As to claim 22, Verma et al. teaches an apparatus that interfaces with a filesystem (see Abstract), comprising:

a processor (see figure 1, element 21);

a memory (see figure 1, element 25) having instructions capable of being executed on the processor…

For the remaining steps of this claim appellant(s) is/are directed to the remarks and discussions made in claim 1 above.

As to claim 23, Verma et al. teaches an apparatus for creating a filesystem with transaction based functionality (see Abstract), comprising:

For the remaining steps of this claim appellant(s) is/are directed to the remarks and discussions made in claim 1 above.

As to claim 24, Verma et al. teaches an apparatus for interfacing with a filesystem (see Abstract), comprising:

For the remaining steps of this claim appellant(s) is/are directed to the remarks and discussions made in claim 1 above.

3. Claim 16 is rejected under 35 U.S.C. 103(a) as being unpatentable over Verma et al., as modified, as applied to claim 12 above, and further in view of Kung et al. ("On optimistic methods for concurrency control", ACM Transactions on Database Systems (TODS), vol. 6, issue 2, pages 213-226. Published June 1981).

As to claim 16, Verma et al., as modified, still does not teach wherein determining the one or more data dependencies includes using optimistic concurrency control (OCC) to control pending read and write operations to the pseudo-filesystem, the filesystem and one or more corresponding files associated with the pseudo-filesystem and filesystem respectively.

Kung et al. teaches wherein determining the one or more data dependencies includes using optimistic concurrency control (OCC) to control pending read and write operations to the pseudo-filesystem, the filesystem and one or more corresponding files associated with the pseudo-filesystem and filesystem respectively (see Abstract).

Therefore, it would have been obvious to one of ordinary skill in the relevant art at the time the invention was made to have modified Verma et al., as modified, by the teaching of Kung et al. for the benefit of providing an external transaction service (See Verma et al., column 6, lines 59-64, where one type of transaction service, MS-DTC, is suggested. Furthermore, Examiner notes that there are 171 citations listed on the ACM Portal, indicating that the method is well-known in the art).

**(10) Response to Argument**

Appellant's arguments presented in the Appeal Brief filed on 31 March 2008 have been fully considered but are not deemed persuasive.

First, it is noted that throughout the Appeal Brief, Appellants refer to themselves as "Applicants". This does not conform with standard practice. After the Notice of

Appeal is filed, "Applicants" become "Appellants". For purposes of this Examiner's Answer, instances of "Applicant" and variations thereof will be read "Appellant".

The Appellant argues "Verma does not teach or suggest duplicating a filesystem into a pseudo-file system and using text based commands to operate on the pseudo-file system". The Examiner respectfully disagrees.

It is noted that Verma is not relied on for teaching the duplication step. Deshayes is relied on for this teaching (see Office Action, 30 October 2007, page 4). Therefore, Appellants' arguments with respect to this limitation are moot.

Verma teaches operating on the pseudo-file system. Verma is directed to a transactional file system that exists on top of a normal file system. A user may choose to operate on files in a transactional mode (see col. 6, ll. 31-55). Because Verma exists on top of a file system, it is a pseudo-file system. Operations on the pseudo-filesystem are taught throughout Verma. If Appellants' assertion that Verma did not teach operating on a file system was true, the invention of Verma would not do anything. A file system (and thus a pseudo-file system) are useful if and only if they support operations on files. For an example of one such operation, see the cited portion of Verma, where committing a modifying transaction is taught (see col. 3, ll. 6-7). A modifying transaction is an operation (i.e. transaction) that changes (i.e. operates on) the underlying file system. To summarize: a user initiates a transaction through the pseudo-file system and this transaction is carried out on the pseudo-file system. This also changes the data in the file system.

At the bottom of page 18, Appellant argues that Verma cannot teach a pseudo-file system because Verma does not teach duplicating a file system in a pseudo-file system. This assertion does not logically follow from Appellant's premise, for the reasons above.

Appellant goes on to argue that the cited portion of Verma "has nothing whatsoever to do with 'operating on one or more portions of the pseudo-filesystem'". Again, the Examiner respectfully disagrees. As stated above, the transactional file system of Verma sits on top of a normal file system and provides additional features that make it a pseudo-file system. At col. 3, ll. 3-6, Verma describes certain features of this pseudo-file system that relate to committing transactions. When transactions are committed, changes are made to the pseudo-file system and are eventually propagated to the underlying file system. The changes constitute "operations."

The Appellant argues that "Berliner does not teach or suggest duplicating a filesystem into a pseudo-file system and using text based commands to operate on the pseudo-file system." The Examiner respectfully disagrees.

Berliner is not relied on for teaching the duplication step. Verma is relied on for this teaching (see Final Rejection, page 4). Therefore, Appellants' arguments with respect to this feature are moot.

Berliner is relied on for teaching the step of receiving text-based commands to operate on the pseudo-file system. The pseudo-file system of Berliner is CVS. CVS is a well-known program that tracks versions of files. These files are stored on an underlying

file system, but managed through CVS. Users can check in (i.e. create or update) and check out (i.e. read) files through CVS. Thus CVS is a pseudo-file system, because it supports operations equivalent to those of a file system but it is not a file system *per se*. Therefore, Berliner and Verma are analogous art.

As stated in the rejection above, section 2.2 of Berliner describes the checkin command. In the Acknowledgements section of Berliner (page 11), the author states that checkin is a "shell script". As is well known in the art, shell scripts are a series of text commands that are typed in by a user, saved, and later executed as if a user entered them by hand at the console (i.e. to an operating system such as Unix, they are identical to commands entered at the console). They provide a shortcut (that is itself a command) for users to combine a series of commonly-used commands into a file that can be executed quickly. (Support for this argument can be found in the document "Running Arbitrary Scripts Under CVS", cited 22 May 2007, which describes writing custom scripts for CVS.)

It is important to note that all the limitation requires is a text command that causes some operation to be performed on the pseudo-file system. In addition to the checkin command script described above, Berliner also supports a join command as described on page 5 below figure 3. Several other commands are described throughout Berliner (e.g. "update", page 3).

The Appellant argues that "Deshayes does not teach or suggest duplicating a filesystem into a pseudo-file system and using text based commands to operate on the pseudo-file system." The Examiner respectfully disagrees.

Deshayes is relied on for teaching the duplicating step. The operating step is taught by Verma as described above. Appellant's arguments with respect to the operating step and Deshayes are moot.

Deshayes is directed towards backing up and restoring virtual partitions. Virtual partitions are pseudo-file systems because they are virtual (i.e. "pseudo") versions of file system elements. "Each separate storage device may be referred to as a 'virtual volume,' or 'virtual disk.' This reflects the operating system's view of the storage device structure may not correspond to the actual physical storage system implementing the structure (hence, 'virtual'). Unlike the application level 10, however, the file system 12 perspective is as if the file system 12 were dealing with raw physical devices or volumes. As far as the file system level is concerned, the virtual volumes may be divided up into 'partitions,' which are continuous segments of storage. These partitions are, in fact, 'virtual' partitions, because the partition may actually be stored across a variety of physical storage segments (e.g., hyper-volumes)." (see Deshayes, col. 2, ll. 41-54) Thus, the virtual partitions of Deshayes are pseudo-file systems and are analogous to the systems of Verma and Berliner.

Appellant asserts that Deshayes does not teach duplicating a file system into a pseudo-file system. However, the Abstract of Deshayes describes backing up and restoring to a virtual partition. Backing up a file system creates a duplicate of the file

system in another location. See, for example, col. 11, ll. 19-30. "At a step 100b, the backup space on the backup storage system is allocated and prepared to receive the information to be backed up from the storage system 52, such as preparing a tape storage unit to receive the backup information. The space may be allocated based on the information in the DDTAB file. Thus, if the DDTAB file lists a set of hyper-volumes (or areas within hyper-volumes), a corresponding amount of space may be allocated on the backup storage system 54. In one embodiment, any redundant information (e.g., RAID parity information) is copied and backed up, as though it were data. In other embodiments, only user data is copied to the backup storage system 54." A physical file system (bits of data that are meaningful to a file system) are duplicated (i.e. backed up) to a pseudo-file system (or hyper-volume, defined as a collection of virtual partitions at col. 2, ll. 51-54).

The Appellant argues that none of the references teach "two filesystems, namely a first filesystem and then a pseudo-filesystem wherein a file modified in the pseudo-filesystem is updated to the filesystem". The Examiner respectfully disagrees.

This argument is addressed above. As described above, each of the systems (particularly those of Berliner and Verma) teach a pseudo-filesystem that sits on top of a traditional filesystem. Changes made through the pseudo-filesystem (such as checkins of Berliner or transaction commits of Verma) result in changes to the underlying file system.

The Appellant argues that none of the references teach "updat[ing] a status file with the pseudo-filesystem with a text-based status result for performing the text-based command and updates performed in the filesystem". The Examiner respectfully disagrees.

As stated in the rejection of claim 8, Verma teaches change logging and page streams at lines 57-59 of column 2. This feature is described in additional detail in the paragraph spanning columns 2-3. The log is used to ensure that changes are fully committed and provide error correction in a catastrophic event such as a power failure. Thus, the log (or status file, because it stores the status of the operation) is used to perform the commands and updates.

The Appellant argues that Deshayes does not teach "mounting the entire copy of the filesystem under the pseudo-filesystem". The Examiner respectfully disagrees.

Filesystem duplication is described above. Therefore this argument hinges on whether or not Deshayes teaches mounting a backed up filesystem (or pseudo-filesystem). As stated in the rejection, this feature is taught in the section of Deshayes that describes "mounting or importing virtual volumes" (see col. 12, ll. 54-58).

**(11) Related Proceeding(s) Appendix**

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

/Mark A. X Radtke/

_____

Mark A. X Radtke

Appeal Conference held on 5 June 2008 at 1 PM. Agreement was reached to proceed to the Board of Appeals and Interferences.

Conferees:

/Christian P. Chace/

Supervisory Patent Examiner, Art Unit 2165

_____

Christian Chace

/James K. Trujillo/

Supervisory Patent Examiner, Art Unit 2169

_____

James Trujillo